Observers, Observability, ReLU and CCSLM

A useful connection between classical control theory and modern neural networks is the idea of **observability**.

A micro-lesson on observability

- An **observer** is a mechanism that tries to infer the internal state of a system from what can be observed at its outputs.
- A system is **observable** if, in principle, the available observations contain enough information to distinguish its possible internal states.
- A simple example:
- Suppose $y = Ax$.
- If A is invertible, then x can be recovered from y.
- If A has a nontrivial nullspace, different x's produce the same y.
- Some information about the original state has therefore become unobservable.
- Observability is therefore closely related to the question:

> **What information about the past state survives in the current state or output?**

ReLU as an observability problem

A ReLU layer can be written

$$y = \mathrm{ReLU}(x) = D(x)x,$$

where $D(x)$ is a diagonal matrix containing zeros and ones.

- Consider one coordinate:

$$y = \max(0, x).$$

- If $y = 0$, we cannot know whether

$$x = -1, \quad x = -100, \quad \text{or} \quad x = -10^6.$$

- The ReLU has deliberately discarded those distinctions.

- Consequently, a ReLU layer is not generally invertible.
- After many layers, information can progressively become unobservable.

For a locally fixed activation pattern, a deep ReLU network has an effective linear mapping

$$J = W_{L-1}D_{L-1} \cdots W_1 D_1 W_0 D_0.$$

- If J loses rank:
- some input directions disappear;
- different inputs become locally indistinguishable;

- an observer looking only at the later representation cannot recover those distinctions.
- Thus one can ask a systems-theory question about a deep network:

How observable is the original input from the current representation?

CCSLM changes the viewpoint

Consider a context-selected linear mapping:

$$y = A_c x.$$

Here:

- x is the **data**.
- c is the **context**.
- c selects the linear transformation.
- The context does not necessarily have to be derived from x.

This gives a useful separation:

$$\boxed{\text{context} \rightarrow \text{computational selection}}$$

$$\boxed{\text{data} \rightarrow \text{linear transformation}}$$

This is different from the usual ReLU situation, where the data simultaneously determines the gating pattern and is itself transformed by that gating.

Observability in CCSLM

For

$$x_{k+1} = A_{c_k} x_k,$$

we can ask two different observability questions:

- **Data observability**
- How much information about x_0 remains recoverable?
- **Context observability**
- Can we determine which mappings A_{c_k} were selected?
- **Joint observability**
- Can we recover information about both the original data and the computational trajectory?

An especially simple case is:

$$x_{k+1} = A_{c_k} x_k$$

with every possible A_c invertible.

- Individual transformations need not destroy information.
- Their composition can remain invertible.
- The system can therefore perform highly context-dependent computation without necessarily throwing away information about the data.

A potentially useful research question

Instead of asking only:

> **How expressive is a deep network?**

we can also ask:

> **How observable is the original input after a sequence of context-selected transformations?**

This suggests a very elementary research programme:

- Start with 2×2 matrices.
- Select between two matrices using one context bit.
- Examine whether the mappings are invertible.
- Concatenate two independently selected mappings.
- Measure rank and information loss.
- Introduce ReLU gating.
- Compare data-controlled selection with separately supplied context.
- Finally study long chains of context-selected mappings.

This provides a bridge from the classical **observer/observability** viewpoint to ReLU networks and, potentially, to **Concatenated Context Selected Linear Mappings (CCSLM)**.

The striking possibility is that **context can control computation without necessarily being allowed to destroy the observability of the data being computed upon.**